Fermi National Accelerator Laboratory

FERMILAB-Conf-88/166

Crossbar Switch Backplane and Its Application*

R. Atac, A. Cook, J. Deppe, M. Fischler,
I. Gaines, D. Husby, T. Nash, T. Pham, and T. Zmuda
Advanced Computer Program
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

E. Eichten, G. Hockney, P. Mackenzie, H. B. Thacker, and D. Toussaint
Theoretical Physics Group
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

November 1988

*Talk presented at the 1988 IEEE Nuclear Science Symposium, Orlando, Florida, November 8-13, 1988.

CROSSBAR SWITCH BACKPLANE AND ITS APPLICATION

R. Atac, A. Cook, J. Deppe, M. Fischler, I. Gaines, D. Husby, T. Nash, T. Pham, and T.Zmuda
Advanced Computer Program
*Fermi National Accelerator Laboratory**
Batavia, IL 60510 USA

E. Eichten, G. Hockney, P. Mackenzie, H. B. Thacker, D. Toussaint
Theoretical Physics Group
Fermi National Accelerator Laboratory
Batavia, IL 60510 USA

Abstract

A crossbar switch backplane design (Bus Switch Backplane) based on TI's crossbar switch chip is described. This backplane holds a maximum of 16 modules and allows simultaneous communications between up to 8 pairs of modules. The aggregate data transfer rate on the backplane is 160 Mbyte/sec. The Bus Switch Backplane is an essential part of the ACP Multi Array Processor, a supercomputer for site oriented problems. The first application of this machine is in Lattice Gauge Theory calculations. The Bus Switch Backplane also finds ready application in data acquisition schemes based on the ACP multimicroprocessor system.

Introduction

There are three major sections of any parallel processing system: processors, memories, and interprocessor communication. Each section is as important as the others. However, processor speed usually determines the requirements for the other two sections. Interprocessor communication is critically important for grid based problems where each processor may have to exchange data with every other processor.

Many different networks can be used for interprocessor communication. The crossbar switch is the simplest and most expensive network (see figure 1-1 below for an explanation of crossbar switches). Crossbar networks expand as n^2 where n is the number of processors. The number of links between processors is given by n(n-1)/2. [1]This means that a 256 port crossbar network would require 32640 links, which is not presently practical.

In the past, up to 4 port crossbar switches have been used as an integral part of other networks such as the cube, omega, shuffle-exchange, baseline ..... etc. These networks do not offer the full interconnectivity and bandwidth that a crossbar network has. A bus offers the same interconnectivity as a crossbar for up to 20 processors, at a large cost in bandwidth. Busses have been used in special parallel processors designed for problems where the processing time is large in comparison to the communication time.[2]

Unfortunately, many grid based problems such as aerodynamic simulations, weather forecasting, mechanical stress analysis, chemical simulations, lattice gauge and image analysis, to name a few, have communication requirements that are as demanding as their processing requirements. Communication is largely local in most grid based problems. Locality means that any individual processor would have to communicate only with other neighboring processors. In addition to this local traffic, some grid based problems have global communication requirements, such as lattice gauge algorithms which require that an FFT be done across the entire grid.

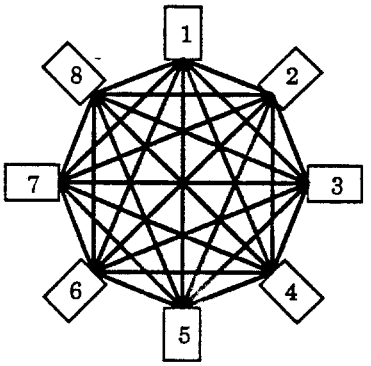

Figure 1-1: This shows the interconnectivity for an 8 port crossbar switch. Each port has a connection to every other port. There are a total of 28 point-to-point connections.

Bus Switch Backplane

A 16 port crossbar switch became economically feasible when Texas Instruments announced their 16 port 4 bit wide crossbar switch chip. Several of these chips can be used together to make 16 port crossbar switches of arbitrary data path width. A new set of network possibilities can be realized using this chip.

At Fermilab's Advanced Computer Program (ACP) group we have created a crossbar switch backplane. It is much like an ordinary bus backplane, which has connectors that processors or other modules can plug into, except the connector signals are not bussed together. Each Crossbar backplane connector's signals connect to one port of the 16 port crossbar switch, which is

* Fermilab is operated by Universities Research Association, Inc., under contract with the U.S. Department of Energy.

resident on the same circuit-board as the connectors. Conversion of a destination address to a crossbar port is handled by a Routing ROM which makes switching topologies quite easy.

Many benefits are gained when the crossbar backplane is used:

1. Up to 8 communications can proceed at the same time.

2. Each communication link operates at 20 Mbyte/sec, giving an aggregate bus bandwidth of 160 Mbytes/sec.

3. Convenience of being able to plug and unplug modules as in a bus like VME.

4. Flexible Routing information gives practically limitless expansion of Bus Switch Backplane networks.

We have designed two modules that plug into this backplane, called the Bus Switch Backplane (BSB). The first module is a Weitek XL chipset based processor which has 10 Mbytes of memory and runs at 10 MIPS and 20MFLOPS. This processor is described in a companion paper called "Floating Point Engine for Lattice Gauge Calculations" written by D. Husby et al. The other module is a BSB to Branch Bus interface module. This module will be described later along with a short description of Branch Bus. In short, Branch Bus is used for inter-backplane communication, and also forms the basis for the communication protocol that the BSB uses.

Crate Based Crossbar Switch

Eurocard specs number IEC-297-3 and IEC-603-2 are used for the Bus Switch Backplane. These specifications are also used by VMEbus and Multibus II. The Eurocard specification allowes the use of commercially available crate hardware, connectors, and module hardware. Figure 2-1 shows a BSB mounted in a crate with modules. The double-board height corresponds to the 6U height (233.4mm) and the single-high board height corresponds to the 3U height (100.0mm). The distance from the back of the board to the front is 280mm.

Branch Bus

Branch Bus is the communications backbone of the ACP's processing systems. It is a high speed cable interconnect used to send and receive blocks of data. Data can be sent in blocks of 1 to 65,536 32-bit words at a maximum data rate of 20 Mbyte/sec. Multiple masters and slaves can reside on a single bus. Once a master has gained control of the bus, it can send/receive several blocks without rearbitration. Branch Bus cables have a length limit of 50 feet. There are Branch Bus interface modules for QBus, UNIBUS, VME, Fastbus, and of course, the BSB. The different Branch Bus modules are listed below:

1. QBBC (QBus to Branch Bus Controller): this module allows a DEC VAX or μVAX to act as a Branch Bus master.

2. BVI (Branch Bus to VMEbus Interface): this module allows a VME bus to act as a Branch Bus slave.

3. VBBC (VMEbus to Branch Bus Controller): allows a module residing on VME bus to act as a master on Branch Bus through the VBBC.

4. FBBC (Fastbus to Branch Bus Controller): allows a module residing on Fastbus to act as a master on Branch Bus through the FBBC.

5. BSIB (Bus Switch Interface Module): this module is the interface between the Bus Switch Backplane and Branch Bus.

Branch Bus protocol is used as the data transmission protocol on the BSB. After a connection is established through the BSB, data is sent as if the data path through the BSB were a Branch Bus cable. Branch Bus in combination with the Bus Switch Backplane forms a powerful communications system for computing systems and data acquisition systems.

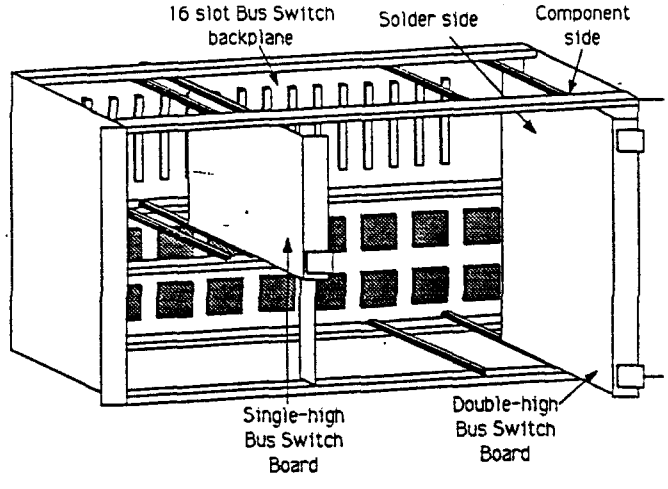

Figure 2-1: Mechanical picture of a Bus Switch Backplane mounted in a Eurocard crate with single high and double high modules. Both the Weitek processor and Branch Bus interconnect are double high modules.

Each BSB has a routing ROM that maps a Branch Bus Address (11 bits) to a crossbar port. This ROM can be configured for different systems. Many different interconnection schemes can be setup. A typical setup would have 8 processors in each BSB and 8 Branch Bus interconnects (BSIBs) that go to other BSBs which have more processors and possibly interconnects to another set of BSBs. This scheme can be thought of as a telephone system for processors, where the BSBs form local city switches and the Branch Bus cables are inter-city trunk lines. Figure 2-2 illustrates this concept.

Bus Switch Operations

There are two sets of signals on each Bus Switch connector: Bus Switch arbitration signals and Branch Bus signals. A module connected to the Bus Switch uses the arbitration signals to open a connection between the port it is connected to and one of the other fifteen ports. The Branch Bus signals are used for data transfer.

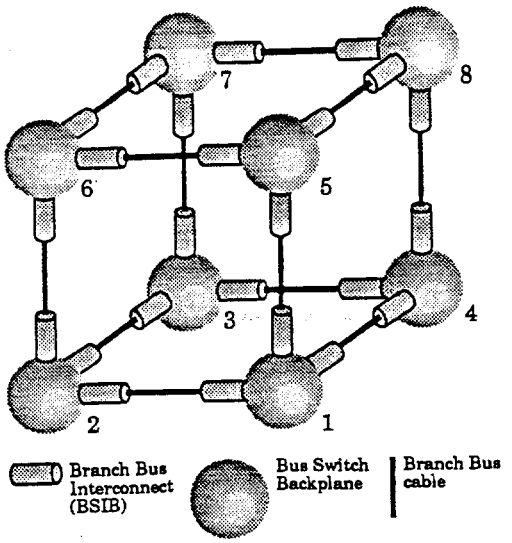

Figure 2-2: Visual representation of a Bus Switch Backplane interconnection architecture. In this example, the BSBs and cable links have been arranged in a cube. (the architecture is not limited to this arrangement) A processor residing in BSB #1 could communicate to a processor in BSB#7 by establishing a link through, for example, BSBs number 2 and 6. The route taken is determined by the routing ROMs resident in each BSB.

During the arbitration, which is when a channel is opened, a module asserts a channel request signal along with the Branch Bus address of the destination module. There are three possible responses a module can get when it requests a connection: destination channel granted, destination channel busy, and an indication that the module requested its own port. It takes approximately 1µsec to open a connection through the switch.

Once a destination channel has been granted, the module can use the Branch Bus signals to start a branch bus transfer between itself and the destination module.

Branch Bus supports the Bus Switch architecture through the use of the BREQ (Bus REQuest) and BGNT (Bus GraNT) signals. Every set of Branch Bus transfers starts when a master asserts BREQ and waits for a BGNT from the destination module.

Figure 2-3 shows a diagram that will be used by the following example. Suppose that module A needs to transfer data to module B. Module A first starts by requesting a channel through the switch. Along with the channel request, the Branch Bus address of module B is given.

The switch processes this request by first using its routing ROM to determine to which port a channel should be opened. The routing ROM indicates slot 8 which has a BSIB module plugged into it. The switch now checks to see if the BSIB is displaying a busy status. If not, the switch will then request use of the BSIB. If the BSIB returns a request granted status the switch will reprogram the crossbar switches and return a channel grant status to module A.

The switch would have returned a channel busy status if there was no module plugged into slot 8, the BSIB was displaying a busy status, or the BSIB did not grant the use of itself. The BSIB arbitrated for use of the Branch Bus cable when the switch requested the BSIB. If the Branch Bus arbitration failed, the BSIB would have returned a busy status.

Module A now asserts BREQ and continues to assert the destination Branch Bus Address (BBA), both of which are received by the BSIB in slot 8. The BSIB then passes the BREQ and BBA onto the Branch Bus cable. The BREQ and BBA are then received by the BSIB module plugged into slot 5 of switch 2. This BSIB module requests a channel through switch 2 and in the process presents the destination BBA it received across the Branch Bus cable.

Switch 2 now goes through a similar procecedure as described above, except that switch 2's routing ROM translates the destination BBA to slot 9 which contains module B. As soon as the channel is open ,module B will receive the BREQ and it will return a BGNT. Once module A receives the BGNT it will start a number of data transfers.

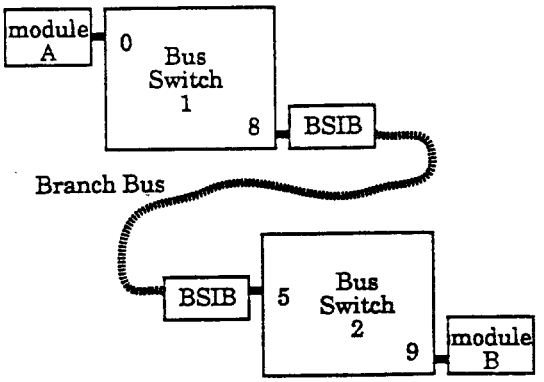

Figure 2-3: Example of BSB operation. A module (A) is plugged into slot 0 of BSB1. There is a Branch Bus cable and two BSIB modules that connect BSB1 to BSB2. The destination module (B) is plugged into slot 9 of BSB2.

Theory of Operation

There are three functions that must be provided by the Bus Switch Backplane: arbitrating between several simultaneous channel requests, programming the crossbar to open a channel, and the actual crossbar function. Each function is served by a different section of the circuit. Arbitration is handled by a round robin arbitration circuit. Crossbar programming is handled by a finite state machine, routing ROM ,and an external configuration register. The crossbar function is taken care of by 13 Texas Instruments 74AS8840 crossbar chips.

Arbiter

One or more modules may assert a channel request signal at the same time. A round robin arbitration circuit arbitrates between these requests and gives a slot number to the finite state machine.This slot number is called the source slot number and is used to route the appropriate source slot signals to the finite state machine. The source slot number is also used to program the external configuration register and ultimately the crossbar chips.

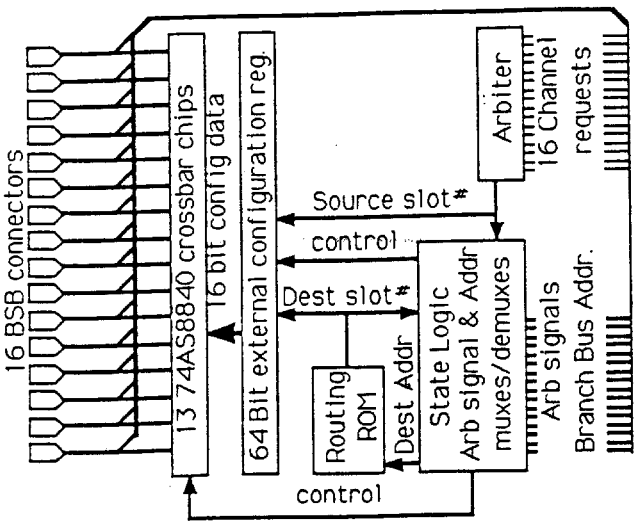

Figure 2-4: Shows a block diagram of the Bus Switch Backplane. Each connector corresponds to a slot on the backplane.

Finite State Machine, Routing ROM, External Configuration Register

Once a channel request has been detected by the arbiter and a source slot number has been generated, the Finite state machine stops the arbiter from allowing any more requests. An 11 bit by 16 channel multiplexer uses the source slot number to select which slot to retrieve the Branch Bus Address. The output of the Branch Bus address multiplexer is connected to the address input of a ROM. This ROM is called the Routing ROM. Its function is to map the Branch Bus Address to a destination slot number. This destination slot is used by the finite state machine to program the external configuration register and ultimately the crossbar chips.

Next the finite state machine checks if the destination slot is busy. If not, it requests the use of the module in that slot and starts programming the external configuration register. The external configuration register has 16 nibbles, each corresponding to one port of the crossbar. Programming is accomplished by writing the source slot number in the nibble corresponding to the destination slot and the writing the destination slot number into the nibble corresponding to the source slot. If the destination module grants use of itself, the external configuration register is written into the crossbar chips.

Crossbar Chips

There are two 64 bit configuration registers inside the crossbar chips: one being used and the other available for programming. The crossbars are programmed by writing the external configuration register into the unused internal configuration register. The internal configuration registers are then swapped and the other register is also programmed with the contents of the external configuration register. The finite state machine then signals to the source and destination module that a channel has been opened and data transfers can commence.

Supercomputer Systems

The topology shown in figure 2-2, can be made into a 64 processor system by placing 8 processors in each BSB crate. More links could be added on all the diagonals of the cube so that any route would have to go over at most one Branch Bus cable link. The maximum bandwidth of the system would be 640 Mbytes/sec (assuming that there are 32 pairs of communications going on simultaneously).

Larger systems can be made by using a hybrid crossbar/hypercube topology. This is the topology that is being used by the Fermilab Multi Array Processor System ACPMAPS. The ACPMAPS machine currently being built is a 256 processor system. It uses 32 BSB crates and 256 interconnect boards (BSIBs) and 128 Branch Bus cable links. This system will be described in more detail later on in the paper.

Data Acquisition Systems

The Bus Switch Backplane was originally designed for the Collider Detector Facility's data acquisition system. It is planned to be used in the third level trigger of their system. The Bus Switch is a unique device that can be used to access an ACP multi processor farm from many different sources. Figure 2-5 illustrates this concept. Data from many sources can be fed to upper level trigger processors while a host machine retrieves good events and places them onto tape. The switch allows data from several different sources to pass to different processors simultaneously resulting in a large communications bandwidth.

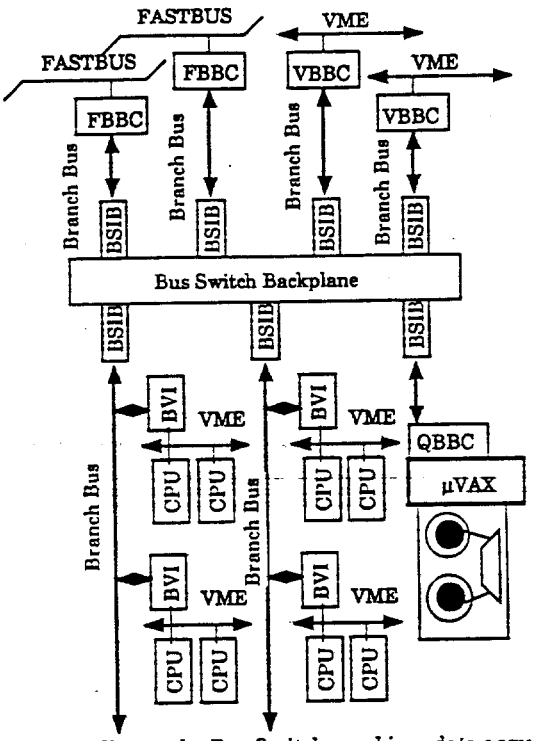

Figure 2-5: Shows the Bus Switch used in a data acquisition system. Data can come from both VME and Fastbus sources and be fed to ACP processors. The ACP processors run higher level triggers and pass good events to the µVAX which stores these events onto tape.

ACP Lattice Gauge Machine

Lattice Gauge is the simulation of quark interactions or Quantum Chromodynamics (QCD). Since QCD is simulated on a four dimensional grid, this is the type of problem which works well on parallel computer that has been described. Communications in QCD simulations range from local communication with grid neighbors to global communications spanning the entire grid.

From analysis of lattice gauge problems, the maximum required interconnect bandwidth between any two BSBs containing 8 processors was found to be about 10 Mbytes/sec, for processors that run at 20 Mflops (peak) and have a data memory size of 8 Mbytes. Branch Bus and the Bus Switch can communicate at twice that rate. See P. Mackenzie [3] or T. Nash [4] for a complete description of this machine.

ACP designed and built a 16 processor system based on the Bus Switch architecture for lattice gauge (called the ACP Multi-Array Processor System ACPMAPS). This project is a collaboration between the ACP Group and Fermilab's Theoretical Physics Group. A 256 processor machine is being built while the 16 processor Cray XMP class machine is being used for code development. A powerful software package called CANOPY automatically distributes the problem as guided by the user. Programs are written in C or FORTRAN as for a single processor and CANOPY will take care of the details of communication and spreading the problem around many processors.

Architecture

Lattice Gauge problems involving dynamical quarks require space-time lattices of at least 32^4 to 64^4, requiring 1 to 16 Gbytes of data memory. The communications requirements are around 1 Gbyte/sec. A 256 processor ACPMAPS system contains 32 Bus Switch Crates, 256 interconnect modules and, of course, 256 processors. A total of 2 Gbytes of data memory, 2 Gbytes/sec interconnect bandwidth and 5 Gflops (peak) are realized with this system. One important note: the processors all run asynchronously - this is a MIMD machine.

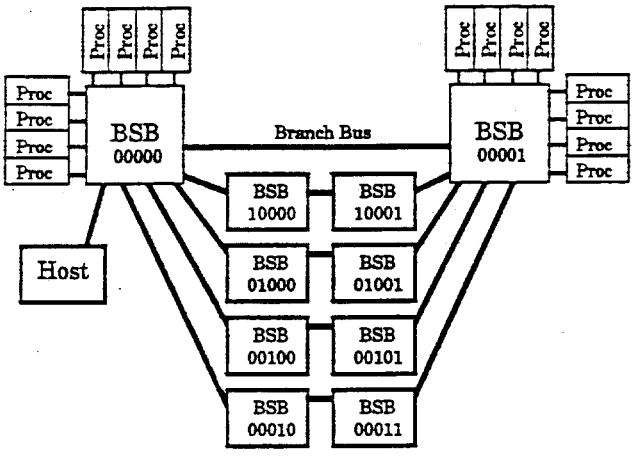

Figure 3-1: A portion of the hypercube topology used for the 256 node ACPMAPS system. Only 10 of the 32 BSBs are shown. Each BSB contains 8 processors.

The 32 BSBs are arranged in a hypercube. Each BSB contains 8 processors and 8 interconnect modules. Figure 3-1 shows a portion of the 256 node ACPMAPS system.

Performance to Date

The following performance data was taken on a 16 node system with 2 BSBs and single Branch Bus link. A monte carlo program, which evolves the link fields on a grid, was run. A 10^4 grid was used for this performance data. The graph below shows the system speedup as processors are added from 2 to 16 processors. In absolute terms, the 16 processor system performs on a suite of lattice gauge calculations at the level of a Cray XMP.

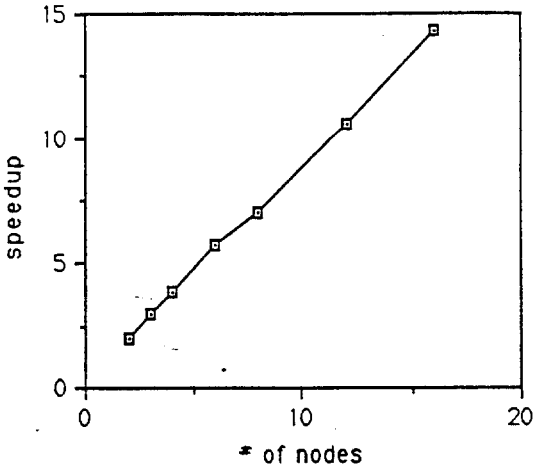

Figure 3-2: Graph of system speedup for 2 to 16 processors on a 10^4 lattice.

Conclusion

A 16 port crossbar switch has been implemented on a backplane. It has many applications including the two that were discussed in this paper. This type of backplane is an excellent way to meet the communication requirements of the future.

References

[1] H. J. Siegel, W. T. Hsu, and M. Jeng, "An Introduction to the Multistage Cube Family of Interconnection Networks," Journal of Supercomputing, vol. 1, no. 1, pp. 13-42, 1987.

[2] T. Nash, et al., "The ACP Multiprocessor System at Fermilab," presented at the XXIII International Conference on High Energy Physics, Berkeley, CA, July 16-23, 1986.

[3] P. Mackenzie, et al., "ACPMAPS: The Fermilab Lattice Supercomputer Project," presented at the International Symposium "Field Theory on the Lattice", Seillac, France, Sept. 28 - Oct. 2, 1987.

[4] T. Nash, et al., "High Performance Parallel Computers for Science: New Developments at the Fermilab Advanced Computer Program," presented at the Workshop on "Computational Atomic and Nuclear Physics at One Gigaflop," Oak Ridge, TN, April 14-16, 1988.